LOOKAHEAD CONVOLUTION LAYER FOR UNIDIRECTIONAL RECURRENT NEURAL NETWORKS

Chong Wang[*], **Dani Yogatama**[*], **Adam Coates, Tony Han, Awni Hannun, Bo Xiao**
Baidu Research, Silicon Valley Artificial Intelligence Lab
Sunnyvale, CA 94089, USA
Contact: `dyogatama@baidu.com`

ABSTRACT

Recurrent neural networks (RNNs) have been shown to be very effective for many sequential prediction problems such as speech recognition, machine translation, part-of-speech tagging, and others. The best variant is typically a bidirectional RNN that learns representation for a sequence by performing a forward and a backward pass through the entire sequence. However, unlike unidirectional RNNs, bidirectional RNNs are challenging to deploy in an online and low-latency setting (e.g., in a speech recognition system), because they need to see an entire sequence before making a prediction. We introduce a lookahead convolution layer that incorporates information from future subsequences in a computationally efficient manner to improve unidirectional recurrent neural networks. We evaluate our method on speech recognition tasks for two languages—English and Chinese. Our experiments show that the proposed method outperforms vanilla unidirectional RNNs and is competitive with bidirectional RNNs in terms of character and word error rates.

1 INTRODUCTION

We are interested in sequential prediction problems, where given an input $\mathbf{x}_{1:T} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\}$, the goal is to make a prediction $y_{1:T} = \{y_1, y_2, \ldots, y_T\}$.[1] In this paper, we will refer to $t = 1, \ldots, T$ as timesteps. Many real-world tasks can be formulated as sequential prediction problems. For example, in speech recognition (language modeling), we are given a spectrogram of power normalized audio clips (a word) at every timestep and predict the character or phoneme (the next word) associated with this input.

Recurrent neural networks (RNNs) are a powerful class of models for sequential prediction problems (Mikolov et al., 2010; Sutskever et al., 2014; Amodei et al., 2015; *inter alia*). There are two general types of RNNs: unidirectional and bidirectional RNNs. Bidirectional RNNs tend to perform better since they incorporate information from future timesteps when making a prediction at timestep t. For bidirectional RNNs, in the forward pass we compute $\mathbf{p}_t = f(b^p + \mathbf{U}^p \mathbf{p}_{t-1} + \mathbf{V}^p \mathbf{x}_t)$, where b^p, $\mathbf{U}^p$, and $\mathbf{V}^p$ are model parameters. Similarly, in the backward pass, we compute $\mathbf{q}_t = f(b^q + \mathbf{U}^q \mathbf{q}_{t+1} + \mathbf{V}^q \mathbf{x}_t)$. The output at timestep t is then computed as $y_t = g(\mathbf{W}[\mathbf{p}_t, \mathbf{q}_t])$, where $[\cdot]$ denotes the vector concatenation operator. For unidirectional RNNs, only the forward pass is performed, so the output at timestep t is $y_t = g(\mathbf{W}\mathbf{p}_t)$. We only consider vanilla recurrent layers in this work, but our technique is compatible with more sophisticated recurrent layers such as LSTMs (Hochreiter & Schmidhuber, 1997) and GRUs (Cho et al., 2014) as well.

Bidirectional RNNs generally achieve better performance since they can incorporate future context, but they come with additional computational costs, both for training and decoding. While an increase in training time is not always an issue (since the training procedure can be carried out offline), an increase in decoding time is a significant issue for a production system that needs to operate in an online, low-latency setting, As can be seen from the equations above, bidirectional RNNs need to wait

[*]Equal contribution.
[1] We use lower case letters to denote variables, bold lower case letters to denote vectors, and bold upper case letters to denote matrices.

for an entire sequence to be seen before making a prediction for timestep t. Unidirectional RNNs, on the other hand, allow decoding in a streaming fashion since they only incorporate previous context.

In this paper, we investigate a computationally efficient way to incorporate information from future timesteps (context) using a new convolution layer. Our goal is to design a method that achieves comparable performance to bidirectional RNNs and still supports online decoding. We show how we can modify a convolutional layer to achieve this purpose in the followings. Our experiments show that our proposed method outperforms vanilla unidirectional RNNs and is competitive with bidirectional RNNs in terms of character and word error rates. This work incorporates new comparisons and discussion not reported in Amodei et al. (2015).

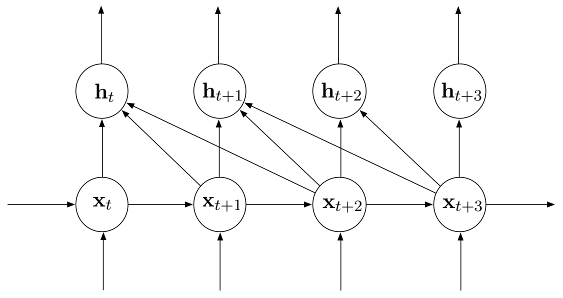

Figure 1: Lookahead convolution architecture with future context size $\tau = 2$.

2 Lookahead Convolution

We propose a convolution layer which we call lookahead convolution, shown in Figure 1. The intuition behind this layer is that we only need a small portion of future information to make an accurate prediction at the current timestep. Suppose at timestep t, we use a future context of τ steps. We now have a feature matrix $\mathbf{X}_{t:t+\tau} = [\mathbf{x}_t, \mathbf{x}_{t+1}, ..., \mathbf{x}_{t+\tau}] \in \mathbb{R}^{d\times(\tau+1)}$. We define a parameter matrix $\mathbf{W} \in \mathbb{R}^{d\times(\tau+1)}$. The activations $\mathbf{h}_t$ for the new layer at time-step t are

$$\mathbf{h}_t = \sum_{j=1}^{\tau+1} \mathbf{w}_j \odot \mathbf{x}_{t+j-1},$$

where $\odot$ denotes an element-wise product. The output at timestep t is then computed as $g(\mathbf{h}_t)$, for a non-linear function g. We note that the convolution-like operation is row oriented for both $\mathbf{W}$ and $\mathbf{X}_{t:t+\tau}$.

3 Experiments

We evaluate our method on speech recognition tasks for two languages: English and Chinese.

Model Our speech recognition system is based on the DeepSpeech system (Amodei et al., 2015). It is a character-level deep recurrent neural network model that takes speech spectrograms as an input and predicts characters at every timestep. Our neural network architecture in these experiments consists of eight layers. The first layer is a regular convolution layer. The next five layers are either all unidirectional (forward) or all bidirectional recurrent layers. The second-to-last layer is the lookahead convolution layer. We also compare with two baselines constructed by replacing the second-to-last layer with either a unidirectional recurrent layer or a bidirectional recurrent layer. The last layer is a softmax layer over character outputs. We train the model using the CTC loss function (Graves et al., 2006). See Amodei et al. (2015) for details of the architecture and training procedure.

Datasets We use the Wall Street Journal corpus[2] for our English experiment and an internal Baidu speech corpus for our Chinese experiment. The WSJ (Baidu) speech corpus consists of approximately 80 (800) hours of training data and 503 (2000) test utterances.

[2] `https://catalog.ldc.upenn.edu/LDC93S6A`

Table 1: Word error rates (English) and character error rates (Chinese) for competing models. We use future context size $\tau = 20$ in all our experiments.

Model	**English**		**Chinese**	
	No LM	**Small LM**	**No LM**	**Small LM**
Forward RNN	23.13	18.79	25.86	15.71
Forward RNN + lookahead-conv	22.66	16.77	21.32	13.45
Bidirectional RNN	19.47	15.42	20.46	12.76

Results Table 1 shows the results for English and Chinese speech recognition. Since our focus is on evaluating the performance of the lookahead convolution layer, we report results without any language model and with a small language model. We note that much better performance can be obtained for both datasets by using a more powerful language model or more training data. We have observed that in both cases the improvements from the lookahead convolution layer are consistent with the smaller scale experiments shown here.

4 DISCUSSION

We showed that the lookahead convolution layer improves unidirectional RNNs for speech recognition on English and Chinese in terms of word and character error rates. We place the lookahead convolution layer above all (unidirectional) recurrent layers. The advantages are twofold. First, this allows us to stream all computations below the lookahead convolution layer. For the lookahead convolution layer, to get an output at timestep t, we only need the input up to $t + \tau$. Second, this results in better performance in our experiments. We conjecture that the recurrent layers have learned good feature representations, so the lookahead convolution layer simply gathers the appropriate future information to feed to the classifier.

We note that there is still a small performance gap between bidirectional RNNs and unidirectional RNNs with lookahead convolution. In our preliminary experiments, we found that increasing future context size did not close this gap. We also found that incorporating future context using a regular convolution layer with multiple filters resulted in poor performance. We obeserved that the resulting model overfit the training data, even after an extensive tuning of the layer hyperparameters. A regular convolution layer also has higher computational complexity than the lookahead convolution layer (although the latency is still lower than a bidirectional recurrent layer). We plan to run more experiments with different future context size and for other sequential prediction tasks to evaluate the effectiveness of the proposed method.

REFERENCES

D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos, E. Elsen, J. Engel, L. Fan, C. Fougner, T. Han, A. Hannun, B. Jun, P. LeGresley, L. Lin, S. Narang, A. Ng, S. Ozair, R. Prenger, J. Raiman, S. Satheesh, D. Seetapun, S. Sengupta, Y. Wang, Z. Wang, C. Wang, B. Xiao, D. Yogatama, J. Zhan, and Z. Zhu. Deep speech 2: End-to-end speech recognition in english and mandarin. *ArXiv e-prints*, 2015.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proc. of EMNLP*, 2014.

Alex Graves, Santiago Fernandez, Faustino Gomez, and Jurgen Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proc. of ICML*, 2006.

Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997.

Tomas Mikolov, Martin Karafiat, Lukas Burget, Jan ”Honza” Cernocky, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Proc. of Interspeech*, 2010.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proc. of NIPS*, 2014.